An Infrastructure for Service Oriented Sensor Networks

Åke Östmark, Jens Eliasson, Per Lindgren
Dept. of Computer Science and Electrical Engineering
Luleå University of Technology, Sweden
Email: {ake.ostmark, jens.eliasson, per.lindgren}@ltu.se

Aart van Halteren, Lianne Meppelink
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente, The Netherlands
Email: {halteren, meppelin}@ewi.utwente.nl

***Abstract*—Emerging wireless technologies enable ubiquitous access to networked services. Integration of wireless technologies into sensor and actuator nodes provides the means for remote access and control. However, ad hoc deployment of nodes complicates the process of finding, selecting and using these in a meaningful way. The use of a service discovery framework enables nodes to present themselves and the resources they hold. In this paper, we review the applicability of a number of well-known service discovery protocols in the context of networked nodes. Multicast DNS and Service Discovery (mDNS-SD) stands out with its auto-configuration, distributed architecture, sharing of resources, and wide area access. For wireless battery operated and resource constrained nodes, we seek to integrate SD and power management techniques. This leads us to a standards based infrastructure for service oriented sensor networks where; 1) nodes collaborate in an ad hoc fashion by using SD techniques to discover (and announce) resources locally and over the public Internet, 2) nodes preserve power through aggressive utilization of low power (sleep) modes, while yet being reachable for clients according to defined schemas, and 3) clients may access and configure nodes, and (if possible) access sleeping nodes by implicit wake-up procedures. To demonstrate the proposed infrastructure a complete experimental setup has been devised featuring; Bluetooth enabled nodes, lightweight implementations of mDNS-SD and communication stacks, Internet access through cellular/wired gateways, together with a public DNS server. Our experiments verify that mDNS-SD can be effectively deployed on small wireless sensor and actuator nodes and provides the basis of a service oriented infrastructure for low power sensor networks.**

***Index Terms*— Service discovery, wireless sensor, sensor networks, activation schedule, low power, ad hoc.**

This paper is based on "Service and Device Discovery of Nodes in a Wireless Sensor Network", by Å. Östmark, P. Lindgren, A. van Halteren, and L. Meppelink which appeared in the Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC06), Las Vegas, USA, January 2006. © 2006 IEEE.

I. INTRODUCTION

Sensors are an integral part of our environment. Smaller chips, emerging wireless communication standards, and more capable sensors and actuators are pushing the development towards wireless sensor networks. Wireless sensor networks are composed of multifunctional miniature sensor devices with limited processing and storage capabilities, that are often battery operated, and interconnected with each other over wireless links. Over the past years, networked sensors have received more attention in various sectors. One application area is a health Body Area Network (BAN), consisting of a network of sensors carried by, and moving around with, the patient. The benefits from mobile (wireless) monitoring in out-of-hospital environments have been the focus in several studies and a number of prototypes have been developed e.g. [9], [10], [11], [35]. These systems incorporate a range of wireless devices with varying capabilities. Even though a small number of nodes might be carried at the same time, the network of sensors may be configured quite differently over time (e.g. depending on how the medical condition of a patient develops over time). Another application area is environmental monitoring where networked sensors monitor various environmental parameters; e.g. water level, temperature, and wind sensors can be deployed along a river bank, issuing automatic warnings in the case of possible flooding [27], [28].

The common basis for the above application areas is the introduction of new sensors, often constructed by various manufacturers. This requires a specific understanding of the type of signal produced by the device and the proprietary protocol for communicating with (and controlling of) the sensors. One such example is a lightweight pulse oximeter sensor, providing either 3 or 375 bytes per second depending on configuration [39].

Furthermore, the deployment of a large number of sensor nodes complicates the process of finding, selecting and using these nodes in a meaningful way. And as

already mentioned, sensor nodes are also often battery operated and there is a trade-off between node power and the cost of replacing the batteries in time. To bring together these issues, we seek an architecture where: (1) nodes can be represented and presented for applications through application oriented interfaces; (2) support of low power operation to prolong sensor nodes' operational lifetime; (3) access to nodes has to be as transparent as possible (even in low power mode) thus enabling remote monitoring and control both in an infrastructureless and in an infrastructure topology to support a wide variety of applications.

We address the first issue by exploring a number of well-known established service discovery protocols, which may be used to manage the complexity of sensor networks. By using a service discovery protocol, it becomes possible to make sensor nodes and sensor data available to an application through application oriented interfaces. Multicast DNS-Service Discovery (mDNS-SD [2]), based on well-known standards, stands out with its auto-configuration, distributed ad hoc architecture, sharing of resources, and potentially lightweight implementation. To demonstrate that an emerging standards-based service discovery protocol actually can be deployed on small sensor nodes, we develop a lightweight mDNS-SD implementation for our prototyping sensor platform MULLE [14].

To reduce power consumption, we utilize low power modes of operations found in existing hardware, e.g. both in sensor node radio and microcontrollers. These are well established techniques to reduce power consumption and hence, prolong a node's operational lifetime [3].

Enabling transparent access is performed by merging node discovery and node power management, and we present a technique for integration of the proposed service discovery protocol with the sensor node's low-power architecture. This technique can seamlessly manage sensor nodes in low power mode, while still allowing users to access the nodes as if they were online.

The motivation for our research activity is the vision of an architecture where nodes collaborate, for example in a medical application where sensor nodes can be used to assist monitoring of patients. Sensor nodes are added in an ad hoc fashion to the BAN and when switched on, they utilize service discovery techniques to instantly discover (and announce) resources to peer nodes. To save energy, nodes enter low power mode but may be reachable for clients according to a defined schema, again found by using service discovery techniques. Access to nodes is either from users in close vicinity by using a Personal Digital Assistant (PDA) or from a remote location using a standard web browser.

This paper is organized as follows. Sec. II presents an overview and related work. An overview of our architecture and our development and prototyping platform is introduced in Sec. III. Sec. IV presents design considerations of our implementation of the Service Discovery Protocol. In Sec. V, an application example is presented. Finally in Sec. VI, the paper is concluded.

II. OVERVIEW AND RELATED WORK

A. Service and Device Discovery

Service discovery protocols enable services and service users to dynamically advertise and find available services in a network. They provide the necessary means to describe services so that the service users can determine if a discovered service matches its requirements, as well as utilize this service.

In general, service discovery protocols defines three entities (software elements) interacting together. A *service manager* holds information about the device and the services it is providing. In the analogy of the client/server paradigm, the service manager can be seen as the server providing a set of services to clients. Continuing the client/server analogy, a client in a service discovery protocol is represented by the *service user* entity, which sends queries for a specific service or device of interest, and selects the most appropriate found. Finally, some service discovery techniques implements a *service cache manager* (an entity tracking all available services) and is the entity where service managers register the services and services user queries for available services. Today, there exist a number of proposed service discovery protocols and the common building blocks and techniques of service discovery protocols include:

(1) Service Catalogues: Service discovery protocols can be categorized as either a centralized directory-based protocol or distributed directory-less protocol. In the former, nodes register their available services with a central repository where service users query for available services. In the latter scheme, the protocol is inherently peer-to-peer and the service catalogue is distributed over the nodes.

(2): Service Description The service discovery protocol must define a data description language, representing and describing the service. In addition, the additional capabilities of the service, or attributes, usually have a standard naming convention.

(3) Registration & Discovery: For service users (clients) to be able to find other nodes and available services, the services must be registered and a discovery process has to take place. To discover services, the discovery process can either be active (by issuing queries) or passive (by listening on service announcements from peer nodes).

(4) Utilization: Other important characteristics for service discovery protocols are the techniques for supporting service delivery and service invocation. For some service discovery protocols, the responsibility for service invocation is controlled by higher level protocols apart from the actual service discovery protocol. Other protocols provide the necessary means to utilize the service by exporting a service interface.

TABLE 1 SHORT OVERVIEW OF SERVICE DISCOVERY PROTOCOLS

	SLP [8]	Jini [21]	UPnP [22]	mDNS-SD [2]
Service catalogues	Centralized or distributed	Centralized	Distributed	Distributed
Capability description	Service templates	Interface and Entry objects	XML device templates	DNS TXT records
Service registration	Unicast to DA or multicast advertisements	Contact lookup service	Multicast advertisements	Multicast advertisements
Service discovery	Unicast to DA or multicast to SA	Query to lookup service	Multicast query	Multicast query
Utilization	Unspecified	Proxy objects	SOAP	Unspecified
Service status	Polling only	Polling or notification	Polling or notification	Polling or notification

(5) Service Status: To maintain a consistent state, it is necessary that the service discovery has a mechanism to notify the service users, ensuring that a clients' knowledge of an announced service is still valid. Either a client can receive a change of a service state by receiving asynchronous notification of a specific event, or by frequently polling the service.

Depending on the application area and usage scenario, the service discovery protocol has a number of requirements. In e.g. pervasive environments, it can be expected that the service discovery protocol must be able to cope with a number of different devices. In such environments, it is anticipated that devices are heterogeneous, ranging from very resource limited tiny nodes to more resource rich devices, e.g. PDAs and laptops carried by human users. For small devices, the processing power, storage capabilities in terms of memory, and communication capabilities must be taken into consideration. In addition, nodes are expected to be in a low-power state with a low duty cycle to conserve power. If the node is not reachable in this state, cooperative techniques must be handled by the service discovery protocol.

B. Well Known Service Discovery Protocols

Well known service discovery protocols include Service Location Protocol (SLP), Jini, UPnP, and mDNS-SD. Table 1 summarizes our comparison of these protocols. The presented service discovery protocols have taken different approaches to enable dynamic service registration, discovery, and service invocation. For example, mDNS-SD and UPnP have a clear focus on enabling address allocation without DHCP servers, automatic discovery of computers, devices, and services on IP-based networks (as known as zero-configuration networking). Furthermore, in Jini, services are delivered as Java objects to service users requesting the service, making it possible to perform ordinary method calls. Service delivery and invocation in SLP and mDNS- SD, is on the other hand entirely left out from the protocol description.

C. Choice of Service Discovery Protocol

The choice of service discovery protocol for our sensor node is based on a number of properties associated with ad hoc sensor networks. Our target platform is a resource constrained sensor node, hence we seek a lightweight solution that enables devices and their services to auto-configure, cooperate, adapt to changes, and to dynamically advertise and find available services in a sensor network. The storage, processing, and communication capabilities precludes some of the mentioned protocols. In such environments, initiatives have been made to support resource-constrained devices, for example the Jini Surrogate Architecture (for devices without a Java Virtual Machine) [20]. In this framework devices may join the service federation with the aid of a surrogate host, a resource rich device, representing and acting on behalf of the non-Jini capable device. On one hand, the Jini Surrogate Architecture may solve the issue of nodes having limited capability, but on the other hand, it enforces clients searching for services provided by the device to apply Java/Jini technology. Others have outsourced large and complex tasks to dedicated and more powerful nodes, allowing small nodes to become a part of an UPnP environment [7]. However, both approaches require a hosting environment that must be provided by a resource rich device. Furthermore, due to intermittent network connectivity in ad hoc environments, nodes may appear and disappear without notification. For an application that needs to maintain a consistent view of the available services, the node may either poll the network repeatedly or receive a notification when a change occurs. As seen in Table 1 Jini, UPnP, and mDNS-SD support the notion of detecting a change of the service status state by either polling or receiving a notification event. SLP relies on polling, but work has been performed to support notification as well to detect changes [26]. In addition, in ad hoc networks with devices acting as routers and hosts at the same time, forming an arbitrary topology, the service discovery protocol must not rely on a centralized architecture (e.g. as Jini presuming the existence of a central repository for service registration and service lookup). Finally, dynamic service discovery protocols are often designed to be

scalable in local networks [8], [15]. Extending from searching and browsing for services in radio proximity, the possibility to register services and perform lookups from the global Internet would be beneficial. SLP and mDNS-SD present solutions to operate in a local scope as well as searching for services in a global scope. This is achieved by utilizing extensions to existing standards (DNS) thus enabling remote service discovery. Since mDNS-SD in particular is based on DNS, whereas SLP adds it for remote wide area service discovery, we have selected the mDNS-SD device and service discovery protocol as being suitable for our implementation on the sensor nodes. This gives the nodes the option (as required by the target application area) of publishing services both in the .local domain as well as in a public domain using Dynamic DNS updates, which enables nodes to register and dynamically update their service records whenever changes occurs.

D. Low Power Operation

Continuing the last section, the research effort related to resource and service registration and lookup in sensor networks has generally been biased towards middleware systems. In such systems, both services and sensor readings may be sent to a central service gateway, collected, and stored for future use by service clients. However, pushing service discovery techniques into the sensor network itself appears to be less explored. One exception to this is found in the area of routing, where network routes in the wireless sensor network are interpreted as services, and where service discovery techniques are used for building energy-efficient network paths. This power saving technique is however not restricted to the network layer. For example scheduled (or duty-cycle based) activation for power efficient MAC layer protocols is a well known and popular technique to minimize the power consumption of sensor nodes (i.e. nodes coordinate their (sleep) schedule to support low duty-cycle operation).

Our approach is similar, using existing low power modes provided by the sensor node and utilizing node scheduling to optimize power consumption. The differences being that: (1) in e.g. [17], [18], [19], the target network is a large number of distributed sensors, interconnected in a multi-hop network and the focus is on developing new MAC layers to address power constraints, whereas we foresee applications in smaller existing single-hop Personal/Body Area Networks; (2) In the mentioned traditional Wireless Sensor Networks, the scheduling is on a node-to-node communication level whereas a PAN involves node-to-node as well as user-to-node communication, and therefore also requires a high level scheduling in order to preserve energy.

E. Transparent Access

The integration of Internet enabled sensor and actuator nodes into measurement systems has been reported before in several cases ([33], [1], [37]). Furthermore, in [29] it is stated that open protocols and a network structure are needed where all nodes are able to conveniently communicate (with each other). To solve the above, access from external Internet and communication within the sensor network, many projects propose conceptually similar solutions. In [30], an IP sensor node to seamlessly integrate a device level network to a management level network is presented. The IP sensors export their functionality (capability and self-description) and a user-friendly name is mapped to a node and the mapping of sensors is made by a central server during system configuration. In [31], a middleware architecture and a network of MicaZ nodes are presented. The sensor nodes broadcast messages, intercepted by a gateway, and forwarded to a home service framework always running at a home gateway. Access to, state change of, and monitoring of nodes are performed by sending the application-level protocol HTTP, and a mapping is performed by the middleware into MicaZ "understandable" packets. Finally, in [32] a service-oriented platform (a sensor node) as well as a middleware architecture hosting the nodes is presented. During boot phase, the node uploads a given (unique) identification, a device driver bundle, and the representation of the nodes' sensor (modeled as a service) is inserted into the middleware system based on the OSGi framework [40]. The physical nodes are stack-based meaning that a number of different heterogeneous sensors and actuators might be connected, and the OSGi framework enables a mechanism for translating the connected sensor (or actuator) into a software service. For an application (external to the networking sensors e.g. from the Internet) accessing the service, a bundle implementing a proxy interface to the middleware is provided.

Although the above examples have different application areas, such as industrial networks, home networks and home appliances, or smart houses, they envision that sensors, actuators, and sensor networks will become a part of the future Internet. As a result, the proposed architectures may solve the issues of discovering resources (locally and remote), allow low power modes of operation, and initiate wake-up procedures to access nodes. However, the operation of the above is based on the following assumptions: (1) although the deployment of nodes might be *ad hoc*, an infrastructure exists for sensor nodes as well as clients to connect to; (2) the development of a large gateway solution, dealing with the communication activities on behalf of the sensors and actuators is more or less *de facto*, which proxies all application interaction thus hides the internal structure of the networked sensors. These systems could well satisfy the established constraints set on the system, but would suffer if future flexibility is needed. Hence, these assumptions may infer limitations on the range of applications that can be supported by networking sensor platforms. One such application example could be decentralizing processing of a thermostatic control-loop to nodes [34], where the nodes communicate with nearby nodes and perform the appropriate action depending on the inputs, without the need of a master controller node.

Thus we seek an architecture where data can be exchanged between nodes without the support from

intermediate nodes and at the same time provides global access to all individual nodes. This would give us the possibility of application control and monitoring of each and every node in the sensor network. However, such an architecture must not preclude the usage of gateway access points, service-oriented backend frameworks etc. if needed by the application environment. In fact, many applications do need a service backend infrastructure, e.g. to store samples of measured data from the sensors into a standard relational database, but the main difference is that the gateway does not need to be the focal point of entrance to the sensor network, converting external standards based protocols to internal non-proprietary protocols.

III. ARCHITECTURE OVERVIEW

We created the system from available commercial off-the-shelf (COTS) components and communication technologies as a base for the service oriented infrastructure for low power sensor networks. As depicted in Fig. 1, the symbolic functionality represents a high-level view of clients' interest of measurements sampled by (and possible control of) the networking nodes. The concrete implementation consists of a number of networking sensor platforms, interconnected over a short range radio link (the license-free industrial, scientific, and medical (ISM) frequency band). Nodes are presented, and their capabilities announced, to local and remote clients using mDNS-SD as previously described. To support low power operation, nodes duty cycle are scheduled and announced to clients. Supporting "anywhere" remote access and control by e.g. a standard web browser (and depending on the application) nodes may connect to a fixed access point attached to e.g. Ethernet, or connect to an appropriate mobile phone for applications requiring enhanced mobility.

A. The MULLE platform

Over the years, a number of prototypes of wireless networking sensor nodes have been developed. Many of these devices are built using COTS components. COTS hardware platforms such as the Berkley Mica motes [12] have often been used when developing applications. Another platform is the BTNode [5], a demonstration and research platform. Others are the already mentioned IP sensor [30] and the Atlas node [32]. Our representation of a node is similar. The major hardware components on our prototyping sensor platform MULLE [14] are a 16-bit microcontroller, a Bluetooth single-chip module with integrated antenna, and an interface to connect sensors and actuators. The software architecture consists of lightweight TCP/IP and Bluetooth stacks. Utilizing standard protocols for communication has several advantages over developing proprietary protocols. For example, it makes it possible for nodes to operate seamlessly with different types of devices (other nodes, access points, mobile phones, PDAs etc). Also, Bluetooth is a widely accepted wireless standard and enables short-range wireless data communication between devices. It has been argued that a wireless sensor node, having the limited computation, memory, and communication

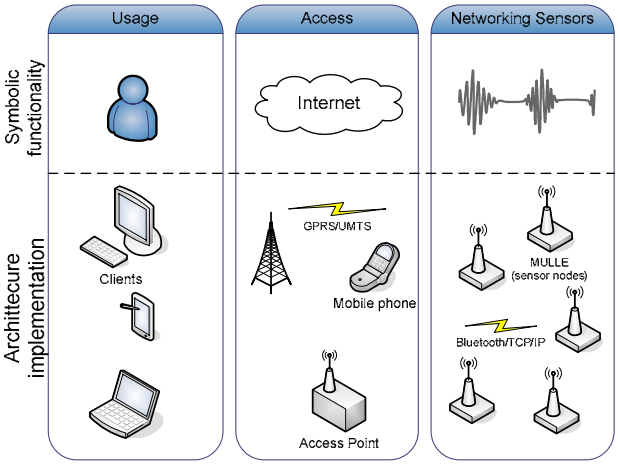

Figure 1. System overview. The symbolic functionality of the system illustrates a user request for remote sensor data, and the architecture implementation represents the concrete implementation.

resources is precluded from the use of the "heavyweight" networking protocols [11], [16]. It is true that the components assembled on the development platform incorporate more powerful components than normally found on small sensor nodes [23], [13]. However, it has been shown that the TCP/IP protocol can be used on COTS devices, similar as those traditionally used as sensor nodes [6]. Finally, achieving interoperability with a large number of devices makes Bluetooth valuable, especially when prototyping sensor-based applications.

B. Operational Modes

One of the most important design issues when it comes to embedded systems is the power consumption. Often, nodes are situated in places that make it difficult and time consuming to change batteries. If nobody is around to replace empty batteries, the sensor node may also cease to operate properly. Therefore, it is vital to monitor the battery capacity, and inform the user of the estimated lifetime. It is also equally important to minimize the power consumption. In most systems, the wireless radio communication is the most power consuming part, and also the one that is the most difficult to minimize.

By using an *activation schedule*, a node can be instructed to transmit its data at certain times and intervals. The schedule allows a user to control how often and how long a sensor node turns on the radio, and therefore gives an easy way of calculating the power consumption. The calculations can thereafter be used to find the required capacity of the battery, or give an indication on how often a node can use the radio given a fixed battery capacity. The operational mode on the MULLE that is based on an activation schedule is called *Time-synchronous*. For an in depth description, all available modes for the MULLE is extensible described in [36], but a brief summary is outlined below:

- *Active mode* Initiates a connection according to (its) requirements; otherwise the radio is turned off. This mode gives low power consumption, but long and non-predictable latency.
- *Passive mode* The radio always on (listening). This allows a low and predictable latency, but relatively high power consumption.

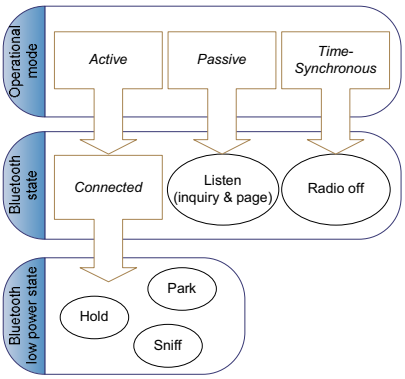

Figure 2. The operational modes mapped to Bluetooth states.

- *Time-synchronous mode* Uses a schedule to switch the radio on (in Active or Passive mode) or completely off.

The Time-synchronous operating mode can have its modes mapped against different low power modes that are supported by the radio technology it is used with. Since the MULLE is Bluetooth based, we mapped Active, Passive, and Time-synchronous against the following Bluetooth states: Connected and Listening. The low power modes in use are; Park state and Sniff mode. This mapping scheme is depicted in Fig 2.

C. Activation Schedule

An activation schedule is maintained by the user, and downloaded into the MULLE node. This schedule controls when and, by using different operational modes, how the MULLE activates its radio. When a node switches off its radio, and hence stops all radio traffic, a user will be notified that the node is no longer available. However, the system will inform the user, with the date and time, when the node will be available next time. The wake-up schedule is described in a calendar file. See Fig. 3 for an example of a vCal-based activation schedule, which instructs the MULLE to go into Passive mode every 20 minutes every day starting the 24th of December 2006. Active mode can also be described in a similar schedule. The MULLE node may also publish the activation schema (a link to the calendar file) using standard techniques for example using the public service _calTalk._tcp.<domain> [38]. Clients uses mDNS-SD to find devices publishing the calendar service, downloads the calendar description using http, or even subscribes to the calendar using webcal. Clients are notified about a change of state of the MULLE node (e.g. Active/Passive) through events, as stated by the calendar file. An example of the simplest case is using the BEGIN:VALARM, ACTION: DISPLAY which displays a message. If other (more advanced) actions should be taken on an event, additional event-programs must be implemented, e.g. using AppleScript on MacOSX. This is possible due to the calendar specification includes the possibility to use attachments containing executable programs. It must be noted though that these types of alarms are subject to any virus or malicious attack that might occur as a result of executing the attachment, hence scripts and software has to be pre-installed (however they can be downloaded from the MULLE node), or in some other way be validated.

```
BEGIN:VCALENDAR
VERSION:2.0
UID: 0001
BEGIN:VEVENT
DTSTART:20061224T100000Z
RRULE:FREQ=DAILY;BYMINUTE=0,20,40
SUMMARY:Passive mode
END:VEVENT
END:VCALENDAR
```

Figure 3. Example of a node's Activation Schedule, expressed in a standard vCalendar file.

When sensor node has its activation schedule, it also needs to keep track on the clock to be able to turn the radio on at the right times. The MULLE uses the Network Time Protocol (NTP) to obtain a correct time and date. The date information is thereafter programmed into its Real-time Clock (RTC). Experimental results in [36] verify that NTP over a Bluetooth network gives satisfactory precision for high granularity scheduling. A MULLE can host both NTP client and server functionality.

D. Access Model

If the MULLE node is in Active mode (radio on, assuming an IP address has been allocated, "online") the activation scheduled as describe in the calendar file can be located and downloaded using standard techniques, which works well both in the .local domain as well as using wide area service discovery. In the .local domain, clients utilizes multicast to find the appropriate services, whereas performing wide area service discover involves querying a Domain Name Server, with records updated by the nodes using Dynamic Updates. However, if the node is in Passive mode (radio on, Bluetooth controller in e.g. inquiry scan sub-state) someone else has to respond/activate and connect to the device. One solution in the .local domain can be by utilizing a Sleep Proxy [2], where a Sleep Proxy service at a peer sensor node can respond to general queries, for example PTR records transferred to the node implementing the Sleep Proxy. Such scheme works well for example, when clients browse for available node services' in the sensor network. However if the client decides to access the service (to get the node name and the mapping between the node name and an IP address) then the Sleep Proxy must "wake-up" the node so it can answer the queries. This access model of the wake-up/respond procedure must also be supported if access is being made from an external network (i.e. not link-local access).

The access model is however further complicated by the policy and the de facto standard utilized by Internet Service Providers. It is not uncommon that a client, accessing the Internet by the use of mobile phones is provided with a private IP address, NATed and hence restricted from running server services accessible from the global Internet. Even though we want the nodes to be a first class entity, where facing three scenarios

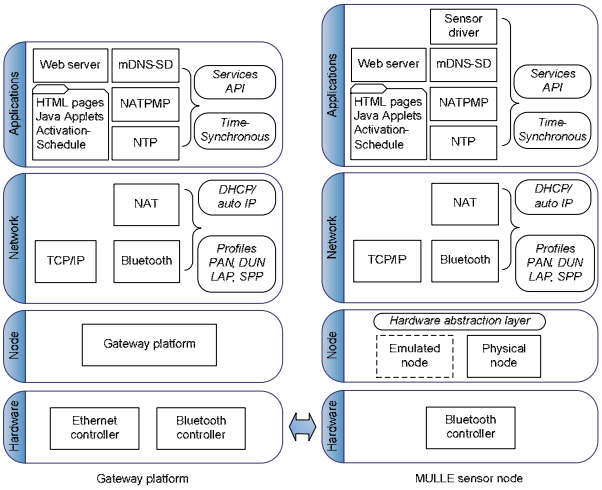

Figure 4. The software architecture implemented on the networking platforms. The right hand side represents the MULLE node, and the left hand side represents a prototype access point platform (connected to Ethernet) respectively.

depending on the access network and policy of telecom operators: (1) *Public IP address allocation*. Nodes are assigned a public IP address and service records are published in the .local domain as well as in a public domain; (2) *Legacy NAT device*. The nodes are assigned a private IP address and outgoing traffic are translated using various techniques (address and/or port number). In this scenario, services may only be announced in the .local domain; (3) *Port forwarding enabled devices.* Various techniques exist to open NAT devices, typically needed by peer-to-peer file-sharing programs. If the device that the nodes connect to (an access point or a peer node) supports this feature, the node may request a port mapping to be performed. In this scenario, we implement NAT-PMP [41] and hence services may be published as in scenario (1) above.

We propose an architecture where nodes are accessible both in Passive mode and Active mode transparently from the clients' point of view. At each node acting as the default gateway for other nodes (typically a Bluetooth Master implementing the Personal Area Networking (PAN) profile), we utilize a Time-Synchronous daemon (TSd). The TSd annotates the Bluetooth connections to slave devices with the current operating mode, mapped from one of the low power modes as describe above. This enables clients transparent, i.e. independent of the nodes operational mode, access to the sensor nodes. For example, a client accessing one of the services provided by one of the nodes (e.g. if we for simplicity assume the activation schedule announces the Active mode), it can not distinguish between the actual Bluetooth state (connected, parked, sniff) mapped by the TSd. What is relevant to the clients is that the node may be accessed, and its services utilized, according to the announced activation schedule. Less important is the actual low power mode activated on the MULLE node; we have to recall though that there is a trade-off between low power consumption and non-predictable latency. This is the case when using e.g. Park state with long beacon interval settings which gives very low power consumption, but high latency.

Preliminary measurements performed on the Bluetooth module, indicates that large power savings can be performed by the proposed architecture, while keeping the total communication delay low and controlled. Depending on the activation schedule, the desired latency can be obtained while keeping the power consumption low. However, more measurements are required to study the overall performance, and its impact on communication latency.

E. Software Architecture

To conclude this section, an overview of the sensor node software architecture is depicted in Fig. 4. As shown, the left-hand side represents the architecture when using an access point with the necessary functionality implemented. For example, the access point may provide nodes with an IP address using DHCP, respond to sensor nodes requests of port mapping (if needed), provide time synchronizing of the node's RTC, responsible for sending a "wake-up" message to parked nodes, and initiating a connection to listening nodes. In the right hand side of the figure, the corresponding software architecture on the sensor nodes is presented. Notably are the similarities between the software architectures, difference being that a sensor node has a sensor driver and a tiny hardware abstraction layer (HAL). The appropriate sensor driver is linked during compile time depending on the attached sensor and target application, and the HAL allows the node software to be executed in emulated mode (e.g. on a PC) during sensor application developing phase. The similarities imply that a sensor node also can act as an access point, providing the services and functionality as described to other networking sensor nodes within radio proximity.

IV. DESIGN CONSIDERATIONS

Our lightweight implementation of mDNS-SD enables the sensor node to announce its services, both in a small network of devices in the .local domain and updating resource records in a DNS. This presentation of services requires the capability of announcing and responding with appropriate DNS messages to wide area clients as well as peer nodes. A significant portion of the DNS messages is used by the representation of domain names. On our platform, the scarcest resource on the node is memory. The 16-bit microcontroller on the platform has 256 kB of flash memory for program code, and 20 kB RAM. Given that in the target environment, almost all information is known at compile time, i.e. the service or set of services, hence we store the appropriate domain names in ROM reducing the amount of dynamic memory needed to be allocated during runtime.

Information not known at compile time, which has to be allocated during boot, is typically the node host name, IP address and the instance part of the mDNS-SD service instance name. The intention of the instance part is to represent a user-friendly name, containing any UTF-8-encoded text. Table 2 shows the size of the code compiled with [24] and executed on the M16C 16-bit

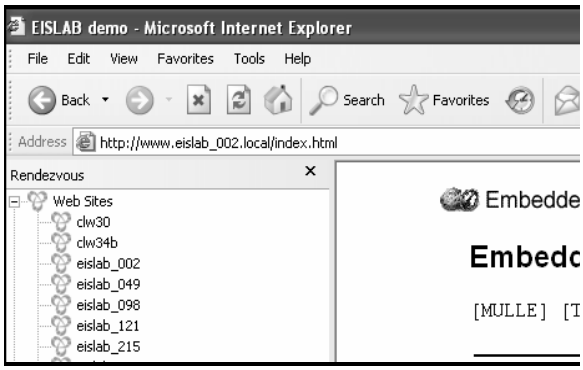

Figure 5. Snapshot of Internet Explorer with mDNS-SD plug-in [4] when connected in a Bluetooth piconet.

microcontroller. In this example, one service is announced by the node. As an indication of the size, we also compiled the code from the mDNS-SD project [25] for the M16C platform and the 32-bit Intel x86-architecture. As an evaluation, those figures cannot be directly compared; the target environments are simply different. For example, the open source code supports much larger DNS messages (up to Ethernet Jumbo frames). However, in our target network technology, such large DNS messages are unlikely to be generated and as a result, we have opted to support the original maximum size of UDP DNS messages (512 bytes). This also gives us an estimation of the memory consumption when processing a query. Even though multiple records might exist in a DNS query message, they are processed one at a time marking potential answers of the nodes services. Assuming the maximum sized DNS message, and the maximum resource record extracted and supported by the node, the processing of queries needs to allocate at most 1 kB of memory during runtime.

V. THEORY OF OPERATION

When the sensor node is switched on, an onboard control application initiates an inquiry to search for other Bluetooth devices in the close proximity. The Bluetooth standard defines a set of profiles for communication. Our platform currently supports the Serial Port Profile (SPP), LAN Access Profile (LAP), the Dial-Up Network (DUN), and the Personal Area Networking Profile (PAN) to access a remote network. The peer device may be a mobile phone with GPRS or a Bluetooth access point, connected to a wired network. During our experiments, we use a gateway prototype platform bridging the Bluetooth and Ethernet networks.

The gateway platform has the same microcontroller as the sensor nodes, Bluetooth hardware, same networking stacks as described in Fig. 4, and is connected to wired Ethernet. The profile role being used is the Network Access Point (NAP), providing a traditional LAN data access point. When connected, a node joins the multicast address assigned for mDNS-SD, and announces its services (both in .local and at a preset DNS). Currently, we experiment with announcing a number of services. The first service is a web-server, announced using the appropriate DNS PTR record with the name _http._tcp.local., and with record data pointing to the specific sensor node. The scenario is depicted in Fig. 5, where the Internet Explorer plug-in is used to show the HTTP services found in the local domain. In this example, a standard PC is connected to a wired Ethernet network as well as connected over Bluetooth acting as a PAN User (PANU) or a Data Terminal (LAP-DT). The plug-in multicast DNS queries for e.g. PTR records to browse the .local domain, and unicast DNS queries to a preset wide area domain, to find out all available web servers, and receives a number of responses. The HTTP service is conveniently utilized, as shown in the figure, by clicking on the appropriate instance found by the lookup process. Other services being elaborated are _ntp (for announcing a Network Time Protocol service) and _calTalk (publishing the node's activation schedule).

TABLE 2. CODE SIZE AND RUNTIME ALLOCATION OF MEMORY WHEN ANNOUNCING ONE SERVICE.

	Lightweight code	MDNS-SD code
Code & constant size	11 kB	132 kB (M16c architecture)
RAM usage, 1 service	329 bytes	11 kB (x86 architecture)

The final service type is the application-specific service, depending on type of sensor attached to the platform. Currently, the abstract service type *'eis'* is used, matching any type of physical sensor connected to the development platform. This usage relies on a higher level application protocol in order to utilize the service i.e. the client application must know the specific details how to access data sampled by the sensor, as well as the format and type of the data. In the case of people accessing the platform by the use of a standard browser, the application protocol could be embedded in a Java applet downloaded from the web server. The use of a generic service type to represent the physical sensor demonstrates one of the strengths with mDNS-SD. In fact, the service discovery protocol supports any application level protocol running on IP based networks.

VI. CONCLUSION

This paper presents the feasibility of the deployment of an emerging lightweight service discovery protocol on wireless sensor network nodes. The target platform is a small sensor node which implements an ad hoc sensor networking device with IP and Bluetooth protocol stacks. We have shown that the nodes are capable of advertising and hosting services both in the .local domain in a Bluetooth piconet and for service browsing in a wide area domain. The lightweight implementation of the device and service discovery protocol is based on well known standards-based communication protocols, thus providing a familiar environment when developing applications for the sensor network. In addition, to reduce power consumption we presented an activation schedule, based on the mapping of the nodes' operational modes to

Bluetooth states. By announcing the activation schedule as a service, a representation of the state of the nodes is exposed to client applications. Finally, an overall architecture is presented incorporating the ideas of device and service discovery techniques and low power operation mode scheduling. Our experiments on a complete testbed featuring sensor nodes, lightweight implementation of mDNS-SD, TCP/IP and Bluetooth stacks, local and wide area access through cellular/wired gateways together with a public DNS server verify that service and device discovery of nodes in an ad hoc sensor network is feasible.

ACKNOWLEDGMENT

The authors would like to thank Kenneth Hartvik for help with various programming issues, and Catherine Man for help with setup of the test environment.

REFERENCES

[1] M. Sveda and R. Vrba, "An integrated framework for Internet-based applications of smart sensors," in *Proceedings of IEEE Sensors*, vol. 2, pp. 1543-1548, 12-14 June 2002.

[2] Apple Computer, Inc., “Rendezvous Technology Brief,“ March 2005. http://www.apple.com/macosx/features/rendezvous/Panther_Rendezvous_TB_10232003.pdf.

[3] V. Raghunathan, C. Schurgers, S. Park and M. B. Srivastava, "Energy-Aware Wireless Microsensor Networks," in *IEEE Signal Processing Magazine*, vol. 19, pp. 40-50, 2002.

[4] Apple Computer, Inc., “Rendezvous for Windows Technology Preview 4,” http://developer.apple.com/macosx/rendezvous/index.html, Last visited: Feb 2005.

[5] J. Beutel, O. Kasten, F. Mattern, K. Roemer, F. Siegemund and L. Thiele, "Prototyping Wireless Sensor Network Applications with BTnodes," in *1st IEEE European Workshop on Wireless Sensor Networks (EWSN), Berlin*, 19-21 Jan 2004.

[6] A. Dunkels, "Full TCP/IP for 8-Bit Architectures," in *Proceedings of the first international conference on mobile applications, systems and services (MOBISYS 2003), San Francisco*, May 2003.

[7] Y. Gsottberger, X. Shi, G. Stromberg, T. Sturm and W. Weber, "Embedding low-cost wireless sensors into universal plug and play environments," in *Wireless Sensor Networks. First European Workshop, EWSN 2004. Proceedings. (Lecture Notes in Comput. Sci. Vol.2920)*, pp. 291 - 306, 2004.

[8] E. Guttman, C. Perkins, J. Veizades and M. Day, "Service Location Protocol, Version 2, RFC 2608," June 1999. IETF.

[9] A.T van Halteren, R. Bults, I.A. Widya, V. Jones, and D. Konstantas, “Mobihealth-Wireless body area networks for healthcare,” in *Wearable eHealth Systems for Personalised Health Management*, Vol. 108 "Studies in Health Technology and Informatics", A. Lymberis and D. de Rossi (Ed.), Il Ciocco Castelvecchio Pascoli Lucca, Tuscany, December 11-14th 2003, ISBN 1 58603 449 9, IOS press, Amsterdam, 2004;

[10] I. Korhonen, J. Parkka and M.V. Gils, "Health monitoring in the home of the future ," in *IEEE Engineering in Medicine and Biology Magazine*, vol. 22, pp. 66-73, May-June 2003.

[11] D. Malan, T. Fulford-Jones, M. Welsh amd S. Moulton, "CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care," in *International Workshop on Wearable and Implantable Body Sensor Networks*, April 2004.

[12] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler and K. Pister, "System Architecture Directions For Networked Sensors," in *Proceedings of the ninth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 93-104, 2000.

[13] J. Hill, M. Horton, R. Kling and L. Krishnamurthy, "The platforms enabling wireless sensor networks," in *Commun. ACM, ACM Press*, vol. 47, pp. 41-46, 2004.

[14] J. Johansson, M. Völker, J. Eliasson, Å. Östmark & P. Lindgren and J. Delsing, "MULLE: A Minimal Sensor Networking Device - Implementation and Manufacturing Challenges," in *Proc. IMAPS Nordic*, pp. 265 - 271, 2004.

[15] C. Lee, A. Helal, N.D.V. Verma and B. Arslan, "Konark: A system and protocols for device independent, peer-to-peer discovery and delivery of mobile services," in *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 33, pp. 682-696, Nov 2003.

[16] J.R.A.H.O. Marcy and A. Twarowski, "Wireless sensor networks for area monitoring and integrated vehicle health management applications," in *wireless sensor networks for area monitoring and integrated vehicle health management applications AIAA Paper 99-4557 AIAA Guidance, Navigation, and Control Conference and Exhibit, Portland, OR, Aug. 9-11, 1999, Collection of Technical Papers. Vol. 1 (A99-36576 09-63)*, 1999.

[17] W. Ye, J. Heidemann and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," in *IEEE/ACM Transactions on Networking*, vol. 12, pp. 493 - 506, June 2004.

[18] T.v. Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *The First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), Los Angeles CA*, Novemeber 2003.

[19] M. Rezaie, V. Mansouri and M. Mani, "Critical area attention in traffic aware dynamic node scheduling for low power sensor networks," in *IEEE Wireless Communications and Networking Conference*, vol. 4, pp. 1933- 1938, 13-17 March 2005.

[20] Sun Microsystems, JiniTM Technology Surrogate Architecture Specification, v1.0 Standard. Oct 2003. http://surrogate.jini.org/sa.pdf.

[21] Sun Microsystems, JiniTM Architecture Specification, Version 2.0. June 2003. http://www.sun.com/software/jini/specs/jini2_0.pdf.

[22] UPnP Forum, UPnPTM Device Architecture v1.0.1 Draft. Dec 2003. http://www.upnp.org/download/UPnPDA10_20000613.htm

[23] M.A.M. Vieira, C.N.C. Jr, D.C.d.S. Junior and J.M. .d. Mata, "Survey on wireless sensor network devices," in *Proceedings IEEE Conference Emerging Technologies and Factory Automation*, vol. 1, pp. 537-544, 19-19 Sept 2003.

[24] IAR Systems. IAR C/EC++ Compiler for M16C V2.12A/W32 (2.12.1.4)

[25] mDNS-SD Developer Web Site, http://developer.apple.com/darwin/projects/mDNS-SD/, Last visited: June 2005.

[26] J. Kempf and J. Goldschmidt, Notification and Subscription for SLP, 3082 March 2001. IETF

[27] S. Tilak, N.B. Abu-Ghazaleh and W. Heinzelman, "A taxonomy of wireless micro-sensor network models," in *SIGMOBILE Mob. Comput. Commun. Rev., ACM Press*, vol. 6, pp. 28-36, 2002.
[28] M. Tubaishat and S. Madria, "Sensor networks: an overview," in *IEEE Potentials*, vol. 22, pp. 20 - 23, April-May 2003.
[29] B. Tao, H. Ding and Y. Xiong, "Design and implementation of an embedded IP sensor for distributed networking sensing," in *Sensors and Actuators A (Physical)*, vol. 119, pp. 567 - 75, April 13 2005.
[30] B. Tao, H. Ding and Y. Xiong, "A novel peer-to-peer distributed sensor network framework based on IP sensor for telemonitoring," in *Assembly Automation*, vol. 26, pp. 137 - 142, 2006.
[31] T. Luckenbach, P. Gober, S. Arbanowski, A. Kotsopoulos and K. Kim, "TinyREST – a Protocol for Integrating Sensor Networks into the Internet," in *Proc. of REALWSN*, 2005.
[32] R.B.J. King and J. Russo, "Atlas: A Service-Oriented Sensor Platform," in *Submitted to the 4th ACM Conference on Embedded Networked Sensor Systems (Sensys), Boulder, Colorado, USA*, vol. 119, 2006.
[33] K. Lee and R. Schneeman, "Internet-based distributed measurement and control applications," in *IEEE Instrumentation & Measurement Magazine*, vol. 2, pp. 23-27, June 1999.
[34] K. Lee and R. Schneeman, "Distributed measurement and control based on the IEEE 1451 smart transducer interface standards," in *IEEE Transactions on Instrumentation and Measurement*, vol. 49, pp. 621-627, June 2000.
[35] C. Kunze, U. Grossmann, W. Stork and K. Muller-Glaser, "Application of Ubiquitous Computing in Personal Health Monitoring Systems," in *Biomedizinische Technik , Band 47, Beitr?ge zur 36. Jahrestagung der Deutschen Gesellschaft fuer Biomedizinische Technik*, pp. 360-362, 2002.
[36] J. Eliasson, M. Lundberg and P. Lindgren, "Time synchronous Bluetooth sensor networks," in *3rd IEEE Consumer Communications and Networking Conference, CCNC 2006*, vol. 1, pp. 336 - 340, 8-10 Jan 2006.
[37] P. Ferrari, A. Flammini, D. Marioli and A. Taroni, "A low-cost Internet-enabled smart sensor," in *Proceedings of IEEE Sensors 2002. First IEEE International Conference on Sensors (Cat. No.02CH37394)*, vol. vol.2, pp. 1549 - 54, 2002.
[38] DNS SRV (RFC 2782) Service Types, http://www.dns-sd.org/ServiceTypes.html
[39] Nonin Medical Inc., 2005. http://www.nonin.com/
[40] C. Lee, D. Nordstedt and S. Helal, "Enabling smart spaces with OSGi," in *IEEE Pervasive Computing*, vol. 2, pp. 89-94, July-Sept 2003.
[41] Cheshire, Krochmal and Sekar, "NAT Port Mapping Protocol (NAT-PMP)," 2005. http://files.dns-sd.org/draft-cheshire-nat-pmp.txt.

Åke Östmark received his M.Sc. in Computer Science at Luleå University of Technology, Luleå, Sweden, in 2002. Since then he has been with the Department of Computer Science and Electrical Engineering (CSEE), Embedded Internet Systems Laboratory (EISLAB), Luleå University of Technology, where he is currently pursuing his Ph.D. degree. His current research interests include embedded systems, Internet enabled networking sensors, and service discovery protocols and architectures.

Jens Eliasson received his M.Sc. in Computer Engineering 2003, at Luleå University of Technology in Sweden. He has since then been working with sensor networks at the Embedded Internet Systems Laboratory (EISLAB). He is now working as a Ph.D.-student; the main research areas involve low power design of Bluetooth-based networked sensor nodes.

Per Lindgren received the M.Sc. in Computer Science at Luleå University of Technology, Sweden in 1994. In 1997 he received the degree of Licentiate in Technology and in 2000 the Ph.D. degree, both in Computer Engineering at Luleå University of Technology. Since 200 he works as a senior lecturer, at the division of EISLAB, Luleå University of Technology. He is currently heading a research group in the area of Computer Engineering with a focus on low power design of Embedded Internet Systems.

Aart T. van Halteren is an assistant professor at the University of Twente. His research interests include software infrastructures for (context-aware) networked applications, body area networks and mobile healthcare applications. As a workpackage manager in the MobiHealth project (IST-2001-36006) he was responsible for the development of a Body Area Network (BAN) for ambulant healthcare. He is currently responsible for the architecture of the MobiHealth platform in multiple collaborative research projects. He holds a Master's degree and a Ph.D. degree in computer science from the University of Twente, The Netherlands.

Lianne Meppelink is studying Telematics at the University of Twente in the Netherlands. She did her bachelor thesis in 2004 at Lulea University of Technology in Sweden. She is especially interested in applications of Telematics and the communication to the end-user.